long Trac

## 1] Franchise Restaurant

(1) To maximize bidding earn:
Accept mile 2, 5, 9 for total of $145000

(2) a. Base Case:

$E(x) \leq 0$ where $x \leq 0$ (because of the 3 mile apart rule, we can go back)

b Rec-Case:

$$E(n) = \max \begin{cases} b(n) + E(n+1) \\ b(n) + E(n-2) \\ b(n) + E(n-3) \end{cases}$$

(3) Pseudo-code

Create an Array E size n and initialize to 0

```
int maxBid (int n, int[] E){
    if E[n] == null {
        if n ≤ 0
            E[n] = 0
        else
            E[n] = b(n) + max(maxBid(n-1, E),
                              maxBid(n-2, E),
                              maxBid(n-3, E));
    }
    return E[n];
}
```

## 2] Inventory Management

(1) Carry 2, 4, 5 because they are the most valuable

(2) Description: Take the max value item in the array V [1 → m] and store that value, Continue to take the max value for the array with the rest of the item in there recursively for n times. Then sum up all the max values we stored

(2) We will proceed this value weight problem by creating a new array that has value per weight unit {2.5, 7.5, 5.67, 5.33, 2.67} and take as much as posible for the highest "value per unit weight" (VPUW)

→ so item2 will be picked first for its VPUW. we can pick total of 3 units of item2 (3 units * 4 weight = 12 weight) so we left with 3 weight left. Next, we look at the 2nd highest VPUW to see if it can fit the left-over weight amount. So item 3 cost 3 weight and it fits just right

→ The algorithm will take 3 item 2
1 item 3
to maximize the value while ensuring the weight limitation

(b) (i) Base Case

$V(n,0) = 0 \quad (n \geq 0)$
$V(n,m) = -\infty \quad (n < 0)$

(ii) Rec-Case

$$V(n,m) = \max\left(V(n - w[m], m-1) + v[m], V(n, m-1)\right)$$

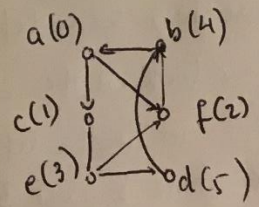
(c) _ create a 2d array V of size n & m & initialize to Null

```
int IMMemRec (int n, int m, int[] V)
    if V[n,m] == null
        if n ≥ 0
            V[n,0] = 0
        else if n < 0
            V[n,m] = -∞
        else
            V[n,m] = max(IMMemRec(n+w[n], m-1, V) + v[m],
                         IMMemRec(n, m-1, V));
    else
        return V[n,m]
```

3] DFS & BFS

(1) BFS on Graph1

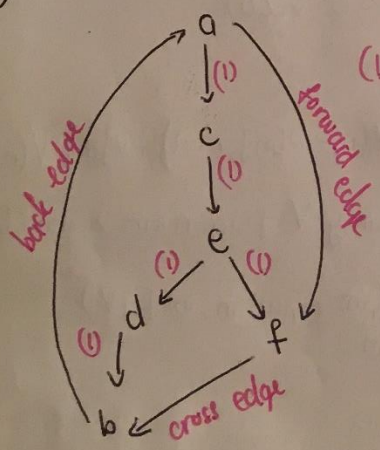
| visit time | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| BFS | a | c | f | e | b | d |




(2) DFS :

| DFS | Visit | End (time) |
|---|---|---|
| a | 0 | 11 |
| b | 4 | 5 |
| c | 1 | 10 |
| d | 3 | 6 |
| e | 2 | 9 |
| f | 7 | 8 |

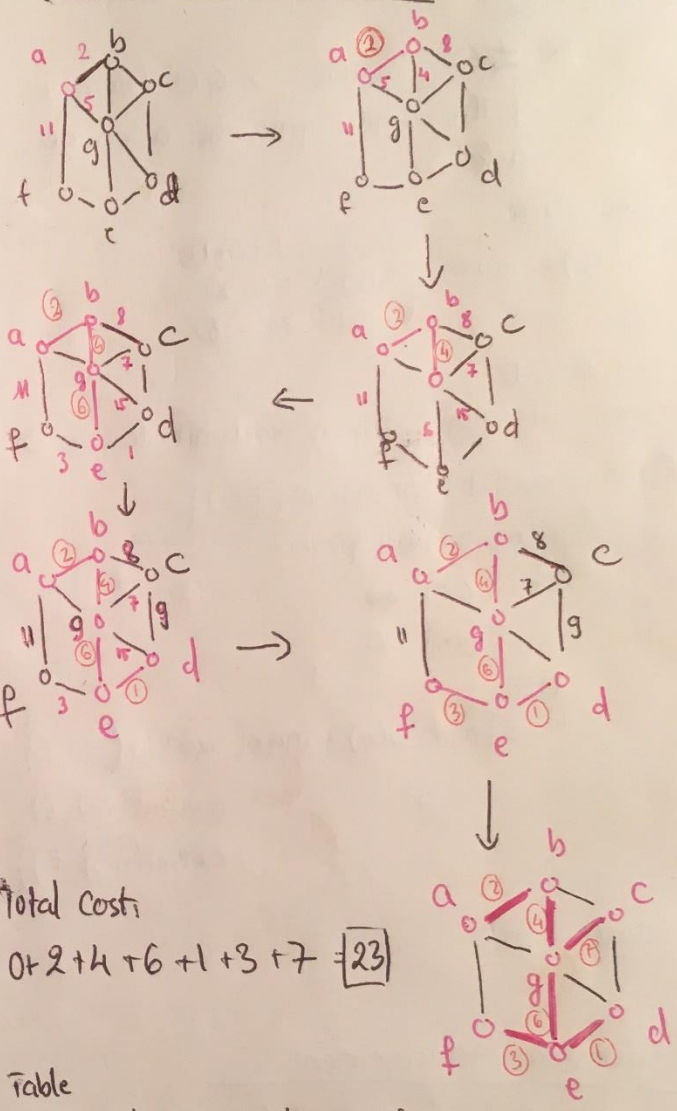
(a)




(1) Tree edge

b) Remove the back edge from b → a

4] Prim's Algorithm



Total Cost:

0+2+4+6+1+3+7 = 23

Table

| | a | b | c | d | e | f | g |
|---|---|---|---|---|---|---|---|
| – | 0 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| a | – | 2 | ∞ | ∞ | ∞ | 11 | 5 |
| b | – | – | 8 | ∞ | ∞ | 11 | 4 |
| g | – | – | 7 | 15 | 6 | 11 | – |
| e | – | – | 7 | 1 | – | 3 | – |
| d | – | – | 7 | – | – | 3 | – |
| f | – | – | 7 | – | – | – | – |
| c | – | – | – | – | – | – | – |